NASA TECHNICAL NOTE

NASA TN D-5384

NASA TN D-5384

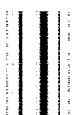
AN ACCURATE STRAPDOWN DIRECTION COSINE ALGORITHM

by John W. Jordan

Electronics Research Center

Cambridge, Mass.

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • SEPTEMBER 1969



1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TN D-5384		

4. Title and Subtitle	5. Report Date
An Accurate Strapdown Direction Cosine Algorithm	September 1969
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
John W. Jordan	C-80

9. Performing Organization Name and Address	10. Work Unit No.
Electronics Research Center Cambridge, Mass. 02139	125-23-02-09 (R-1501)
	11. Contract or Grant No.
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Technical Note
National Aeronautics and Space Administration Washington, D.C. 20546	14. Sponsoring Agency Code

15. Supplementary Notes

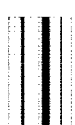
16. Abstract

This report presents an efficient algorithm for calculating the direction cosines of a strapdown inertial system. It is suitable for calculation on a general purpose digital computer; it includes a correction for commutativity error, and has no truncation error. Simulation results are included.

17. Key Words	18. Distribution Statement
•Efficient Algorithm •Direction Cosines •Strapdown Inertial System •Commutativity Error •Truncation Error	Unclassified - Unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price *
Unclassified	Unclassified	15	$3.00

*For sale by the Clearinghouse for Federal Scientific and Technical Information
Springfield, Virginia 22151



AN ACCURATE STRAPDOWN DIRECTION COSINE ALGORITHM

By John W. Jordan
Electronics Research Center

SUMMARY

This report presents an algorithm for computing the direction cosine matrix of a strapdown inertial system. It is designed for efficient computation on a general purpose digital computer. It includes a correction for commutativity error and has no truncation error. Its high accuracy is demonstrated by simulation. The simulation technique can be applied to the evaluation of other direction cosine algorithms as well, permitting a comparative evaluation.

INTRODUCTION

Strapdown inertial systems involve two computational processes which are not present in conventional inertial systems employing a gimbal structure. One is associated with the transformation of vehicle acceleration from body coordinates to navigational coordinates:

$$a_N(t) = C^T(t)\ a_B(t) \tag{1}$$

where

a_B = vector acceleration of the vehicle in body coordinates

a_N = vector acceleration of the vehicle in navigational coordinates

C = the 3x3 matrix of direction cosines which represent the transformation from body to navigational coordinates.

The second process is the determination of the direction cosine matrix by the real-time integration of the matrix differential equation:

$$\dot{C}(t) = \Omega(t)C(t) \tag{2}$$

where $\Omega(t)$ is a skew-symmetric matrix of vehicle rates measured in body coordinates:

$$\Omega(t) = \begin{bmatrix} 0 & W_z(t) & -W_y(t) \\ -W_z(t) & 0 & W_x(t) \\ W_y(t) & -W_x(t) & 0 \end{bmatrix}$$

Both Eqs. (1) and (2) can be solved by a general-purpose computer if they are replaced by finite difference equations. The frequency with which the equations must be solved is determined by a system accuracy specification and the anticipated environment; that is, by the vehicle acceleration and angular rate profile. Generally, the solution of Eq. (1) is not too demanding of the available computational resources (ref. 1). Eq. (2), on the other hand, may require a substantial percentage of the computer's time because of the inherent complexity of the numerical methods used and the high iteration rates which may be required to achieve the specified accuracy. This report is concerned with the solution of Eq. (2) on a general-purpose digital computer. In general, the airborne computer will calculate a matrix B(t) which only approximates the true cosine matrix C(t).

FINITE DIFFERENCE EQUATION

The computer will employ a finite difference equation for propagating the cosine matrix:

$$B^* = \Phi B \tag{3}$$

where

B^* is the new solution at step n

B is the old solution at step n-1.

A suitable transition matrix is given in reference 2:

$$\Phi = I + k_1\theta + k_2\theta^2 \tag{4}$$

where θ is the skew-symmetric rotation matrix:

$$\theta = \begin{bmatrix} 0 & \phi_z & -\phi_y \\ -\phi_z & 0 & \phi_x \\ \phi_y & -\phi_x & 0 \end{bmatrix} \tag{5}$$

Let:

$$\phi_o^2 = \phi_x^2 + \phi_y^2 + \phi_z^2.$$

The scalar parameters k_1 and k_3 are given in Table I along with a scalar k_3 which will be used subsequently.

TABLE I

k_1	k_2	k_3
$\frac{\sin \phi_o}{\phi_o}$	$\frac{1 - \cos \phi_o}{\phi_o^2}$	$\frac{\tan (\phi_o/2)}{\phi_o}$

Equation (4) has appeared frequently in the literature (refs. 1, 4, 5, 7) as an analytical solution of the differential Eq. (2) for a special form of the vehicle rate matrix $\Omega(t)$. For that particular case, the rotation matrix (Eq. (5)) is simply the integral of $\Omega(t)$. In this report, Eq. (4) is used in a different way than in the previous literature. It is presented as a general, exact expression for describing the rotation of a cosine matrix. The problem then becomes the determination of an appropriate rate matrix (Eq. (5)), so that the difference equation (Eqs. (3) and (4)) represents the solution of the direction cosine problem. In general, the rotation matrix is no longer equal to the integral of $\Omega(t)$, although it must, of course, reduce to this form for the special case when Eq. (4) does represent the solution of Eq. (2).

Before considering the determination of the required rotation matrix, Eqs. (3) and (4) will be put in a form suitable for efficient computation on a general=purpose computer.

COMPUTATIONAL ALGORITHM

An efficient computational algorithm may be devised by expressing the matricies in terms of vectors. The direction cosine matrix requires three vectors.

$$B = \left[\begin{array}{c:c:c} b_1 & b_2 & b_3 \end{array}\right]$$

The rotation matrix, being skew-symmetric, can be represented by one vector.

$$\phi = \begin{bmatrix} \phi_x \\ \phi_y \\ \phi_z \end{bmatrix}$$

The computational algorithm is as follows:

$$\phi_o^2 = \phi \cdot \phi \tag{6}$$

Compute k_1, k_3

$$\phi_1 = k_1 \phi$$

$$\phi_3 = k_3 \phi$$

for $j = 1,2$

$$\alpha_j = b_j \times \phi_1$$

$$\beta_j = \alpha_j \times \phi_3$$

$$b_j^* = b_j + (\alpha_j + \beta_j)$$

$$b_3^* = b_1^* \times b_2^*$$

The last step is based upon the fact that the direction cosine matrix is orthogonal. The calculation of k_1 and k_3 is discussed in an Appendix. The following section will discuss the determination of the rotation vector ϕ.

DETERMINING THE ROTATION VECTOR

Strapdown inertial systems measure vehicle rotation by gyroscopes mounted ("strapped down") to the frame of the vehicle. Reference 3 provides an expression for the rotation vector in terms of the integral of the body rates sensed by the gyroscopes

$$\phi = \int_o^t \omega(\lambda)\, d\lambda + a \tag{7}$$

where

ϕ = The rotation vector

ω = The vehicle rate vector sensed by the system gyroscopes

a = A vector equal to the area traced out by each body axis on a unit sphere.

Most inertial systems use rate-integrating gyroscopes which provide the integral of the vehicle rates:

$$\Delta = \int_o^h \omega(\lambda)\, d\lambda$$

The quantity Δ is the vector output of the system gyroscopes over a sampling interval h.

The area vector, a, is found (ref. 3) by applying the two dimensional Green's theorem:

$$a = -\frac{1}{2}\int_o^h \phi \times \dot{\phi}\, d\lambda$$

Then Eq. (7) becomes

$$\Delta = \phi + \frac{1}{2}\int_o^h \phi \times \dot{\phi}\, d\lambda \qquad (8)$$

The problem now is to determine ϕ from the gyroscope output vector Δ. Figure 1 illustrates the idea of a pre-processor. The gyroscopes provide the vector, Δ, which is sampled at a frequency f_s. The pre-processor calculates ϕ from Δ and uses ϕ to update the direction cosines at a frequency f_c. The pre-processor algorithm has two constraints - it must provide sufficient accuracy and it must be computationally simple.

PRE-PROCESSOR ALGORITHM

One possible pre-processor algorithm can be obtained from Eq. (8) by considering the cross-product term to be small compared to the other right-hand term. This is equivalent to choosing the sampling frequency, f_s, high enough so that the direction of the rotation vector changes little over the sampling period.

This assumption results in the approximation

$$\phi \cong \Delta - \frac{1}{2}\int_{o}^{h} \Delta x \omega \, d\lambda \tag{9}$$

In order to evaluate the integral, it is necessary to obtain the vehicle rate vector, ω, from its integral, Δ, which is provided by the system gyroscopes. This can be done by fitting a polynomial to a sequence of Δ vectors and then analytically differentiating the polynomial to obtain the vector ω. Reference 3 introduced the concept of a pre-processor and a polynomial approximation in order to obtain an estimate of the vehicle angular rate matrix, $\Omega(t)$, given a sequence of gyroscope outputs. The angular rate matrix was then used to integrate Eq. (2) directly, using a Runge-Kutta algorithm. In this report, the pre-processor and polynomial approximation are used to determine the equivalent Euler angle rotation from the gyroscope outputs. Reference 6 discusses the use of a pre-processor to provide increased accuracy for direction cosines calculated by the method of quaternions. However, the pre-processor algorithm is equivalent to Eq. (11) of this note; the more general form of Eq. (10) is not derived or simulated. Reference 5 also derives a correction term equal to Eq. (11) by a completely different method.

Let

$$\Delta(\lambda) = d_0 + d_1\lambda + d_2\lambda^2$$

with $\phi = d_0$ being the accumulated value of the rotation vector since the last calculation of the direction cosines. The two remaining coefficients can be determined by two gyroscope samples Δ_1 and Δ_2. See Figure 2. The pre-processor algorithm is then determined by substution to be

$$\psi = \Delta_1 + \Delta_2 \tag{10}$$

$$\phi^* = \phi + \psi + \frac{1}{2}\,\phi \times \psi + \frac{2}{3}\,(\Delta_1 \times \Delta_2)$$

The vector, ϕ, is updated every two gyro samples or at a frequency $f_s/2$. This high-speed calculation is continued until the direction cosines are updated at a lower frequency, f_c. The vector ϕ is then set equal to zero and the high-speed calculations are resumed. For the case where $f_c = f_s/2$, the initial value of ϕ will always be zero and the pre-processor algonithm is simply

$$\phi^* = \Delta_1 + \Delta_2 + \frac{2}{3}\,(\Delta_1 \times \Delta_2) \tag{11}$$

COSINE ERRORS

For some applications, a simplified form of Eq. (4) is sufficient. Approximations for k_1 and k_3 can be used instead of the true values of Table I. The resultant error may be called a truncation error. The truncation error is completely under the control of the designer and it will vanish when the calculations of k_1 and k_3 are made exact. For the remainder of this note, it will be assumed that the accurate values of Table I are used and that there is no truncation error involved in updating the direction cosines. However, the Appendix will consider truncation error resulting from the approximate calculation of k_1 and k_3.

There is another error source which results in what is often called commutativity error. For the algorithm presented in this note, it is caused by the approximate nature of the pre-processor algorithm. In general, the pre-processor provides an estimate $\hat{\phi}$ of the required rotation vector ϕ. Although the commutativity error could be defined as the difference between the vectors ϕ and $\hat{\phi}$, in this note it will be defined as the error in the direction cosines which results from the use of $\hat{\phi}$ instead of the truerotation vector ϕ. See Figure 3. One special case is important. If

$$\phi \times \dot{\phi} = 0$$

then Eq. (8) reduces to

$$\phi = \Delta$$

and the pre-processor is error free. This condition can also be expressed as

$$\omega \times \Delta = 0 \tag{12}$$

which is satisfied if the vehicle rate vector, ω, has a fixed axis in space over the sampling interval. If this condition is satisfied, there is no commutativity error and the direction cosines can be determined exactly. When Eq. (12) is satisfied, the vehicle rate matrix, $\Omega(t)$, corresponding to the vector ω is said to be "self-commutative" (ref. 5). For this special case, the rotation vector is simply the gyroscope output vector.

SIMULATION

The effectiveness of the pre-processor can be demonstrated by simulation. In order to do this, it is necessary to have an analytical solution for the direction cosines with a non-self-

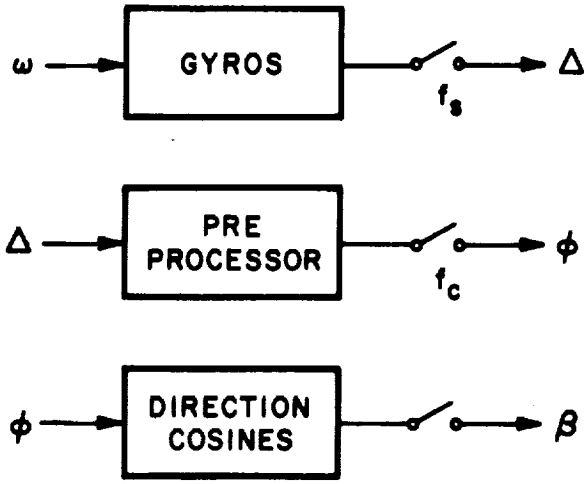


Figure 1.- Pre-processor

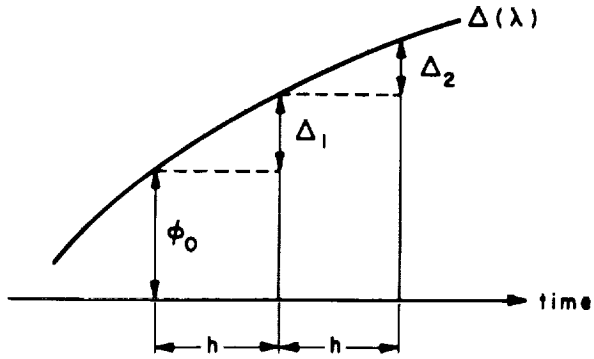


Figure 2.- Gyro samples

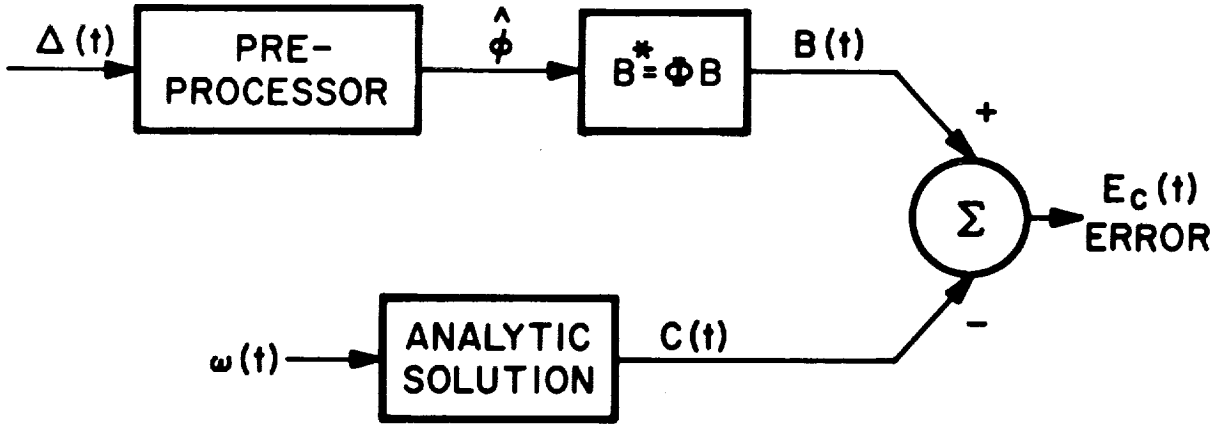


Figure 3.- Error definition

commutative rate matrix. Although no general solution is known, a class of test solutions are easily constructed as follows. Let

$$C = C_A C_B$$

where

$$\dot{C} = \Omega C$$

$$\dot{C}_A = \Omega_A C_A$$

$$\dot{C}_B = \Omega_B C_B$$

If Ω_A and Ω_B are self-commutative, then C_A, C_B and hence C are easily found. The angular rate vector for C is easily shown to be

$$\omega = \omega_A + C_A \omega_B$$

and

$$\Delta = \Delta_A + \int_t^{t+h} C_A \omega_B \, d\lambda$$

Next, ω_A and ω_B are chosen so that the integration can be done analytically. A simple choice of

$$\omega_A = \begin{bmatrix} .6 \\ 0 \\ 0 \end{bmatrix} \qquad \omega_B = \begin{bmatrix} 0 \\ .6 \\ 0 \end{bmatrix}$$

was used. This gives a vehicle rate vector of

$$\omega = \begin{bmatrix} .6 \\ .6 \cos .6t \\ -.6 \sin .6t \end{bmatrix},$$

a severe coning motion. The commutativity error of Figure 3 can be expressed as a single scalar parameter (ref. (4).

$$e_c = k \left[\text{trace } (E_c{}^T E_c) \right]^{\frac{1}{2}}$$

where $k = (57.3)(60)/t_s$

t_s = simulation time (min)

This error parameter has the units of degrees per hour. A value of $t_s = 20$ minutes was used for all simulations. An additional parameter N was defined by

$$f_s = N f_c$$

so that N is the number of gyro samples before the direction cosines are updated. The results for N = 1 correspond to a direct application of Eq. (6) everytime the gyros are sampled without using a pre-processor. Since the accurate values of k_1 and k_3 of Table I were used, none of the algorithms have any truncation error. The results are plotted as a function of f_s and f_c in Figure 4 and 5. Since updating the direction cosines involves the bulk of the computation, the plot of the commutativity error vs. f_c most closely approximates a plot of error vs. computer loading. All algorithms except N = 1 involve the calculation of the pre-processor every two gyro samples. For larger values of N, this requires proportionally more computation and this fact should be kept in mind when interpreting Figure 5. The solution with N = 2 has an additional advantage since its pre-processor algorithm (Eq. (11)) is less complex than that of the others (Eq. (10)).

CONCLUSION

The pre-processor significantly reduces the secular commutativity error with the algorithm N = 2 providing the most accuracy for an equivalent computer loading. Eqs. (6) and (11) provide a high-accuracy algorithm for general-purpose computers which has low commutativity error and no truncation error.

Since the truncation error is easily analysed (Appendix A) and is completely under the control of the designer, it follows that the commutativity error represents a more significant test of an algorithm's preformance. Quite possibly, a more effective pre-processor can be designed, perhaps by using a higher order polynomial approximation. However, it is not the inherent accuracy of an algorithm which is important but rather its realitive effectiveness at a specified level of computer loading.

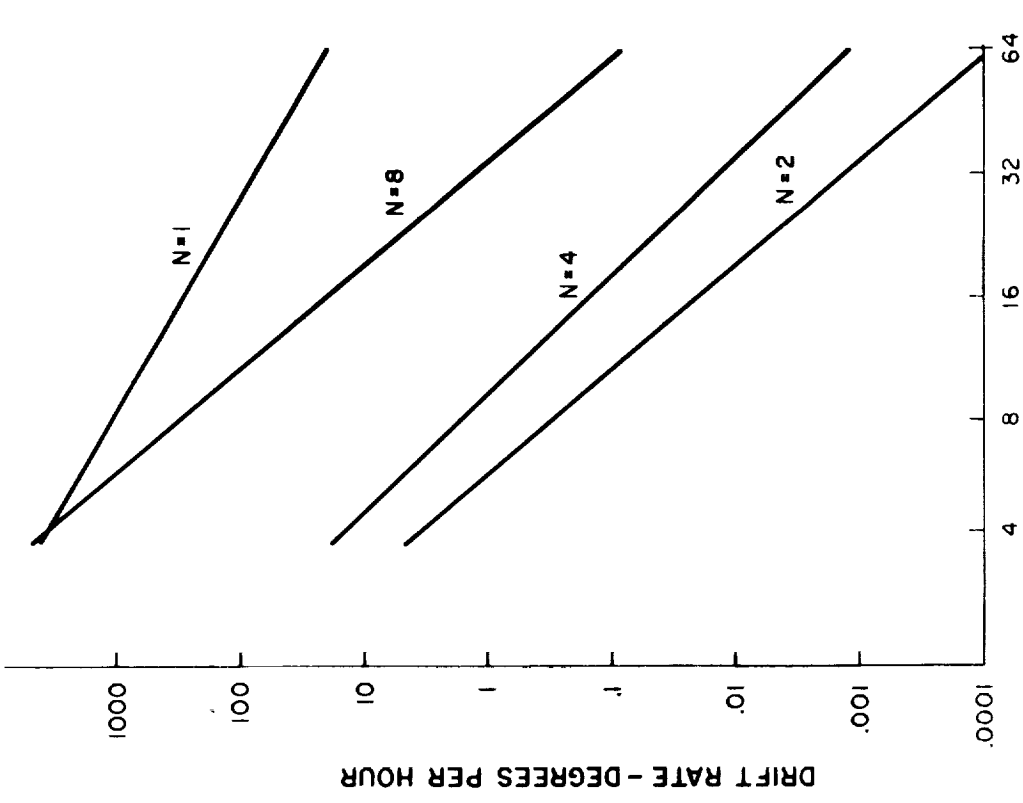


Figure 4.- Commutativity error

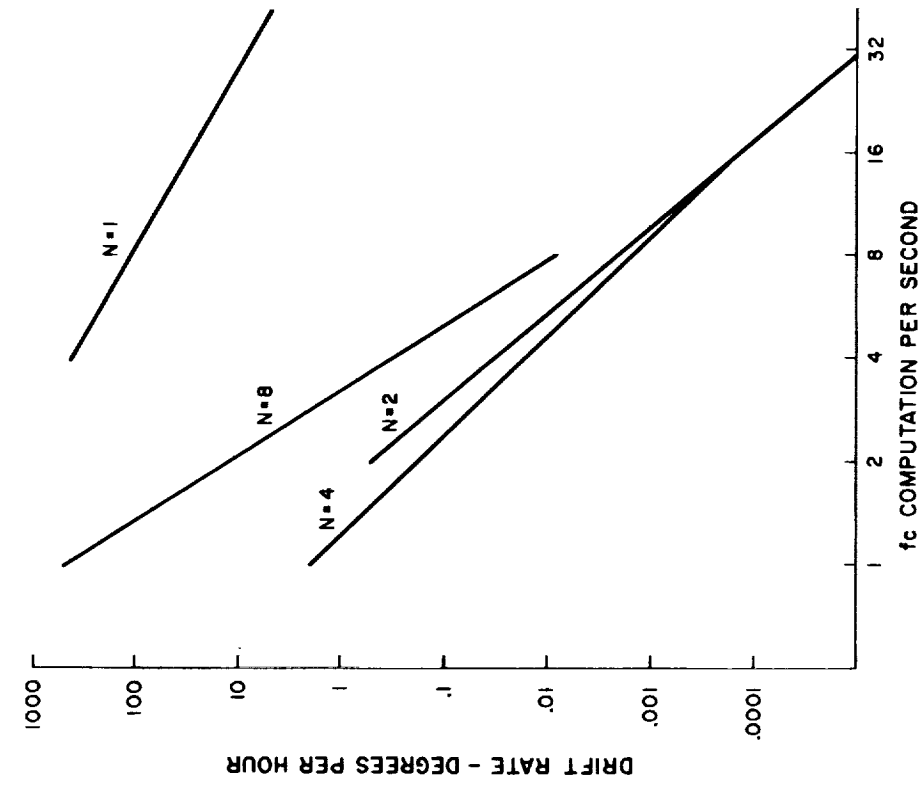


Figure 5.- Commutativity error

APPENDIX A

TRUNCATION ERROR

Truncation error results from approximations made in calculating k_1 and k_3. This error may also be determined by simulation as in Figure 3. For a self-commutative rate matrix, ref. (5) provides an analytical solution for the truncation error. If the approximations for k_1 and k_3 are of low accuracy, the resulting cosine matrix may become non-orthogonal and an orthogonality correction can be introduced. The correction, however, is complex and the added computational time would be more advantageously used in improving the accuracy of k_1 and k_3. Reference 5 contains an analytical solution for the truncation error when an orthogonality correction is used. It also considers the error introduced by finite computer word length. Refs. (1) and (7) also contain an analysis of these error sources.

CALCULATION OF k_1, k_3

The calculation of k_1 and k_3 may be done using standard proceedures for trigonometric functions. However, the expressions in Table I cannot be used since they may require division by zero. For some applications a very simple approximation may be sufficient. For example, refs (1), (5), (6), and (7) discuss $k_1 = 1$, $k_2 = 0$, 1/2, and 1/3. However, by using the pre-processor to correct the commutativity error and by calculating more accurate values of k_1 and k_3, all the calculations can be done at a lower frequency. An excellent discussion of the calculation of trigonometric functions is contained in ref. (8). What follows is based on that reference.

The calculation of $\tan(\chi/2)$ provides a good basis for all the trig functions. For example:

$$\sin \chi = \frac{\tan(\chi/2)}{1 + \tan^2(\chi/2)}$$

Hence

$$k_1 = \frac{k_3}{1 + \phi_0^2 k_3^2}$$

and it remains to calculate k_3. Let $P_n(\chi)$ be the polynomial

$$P_n(\chi) = \sum_{y=o}^{n} a_j \chi^j$$

The tangent function can be calculated as

$$\tan(\pi\chi) = \chi P_n(\chi^2)$$

with the coefficients for a nth order approximation given in ref. (8). In order to determine k_3, only $P_n(\chi^2)$ is necessary. This will be a polynomial in $\phi_0{}^2$, which is provided in Eq. (6). The polynomial may be calculated by Horner's nested form

$$V_n = a_n$$

$$V_k = \chi^2 V_{k+1} + a_k \qquad k = n-1, n-2, \cdots 0$$

$$P_n(\chi^2) = V_o$$

which requires n additions and n multiplications, Reference 8 discusses a streamlined form as well, which requires only 3 multiplications and 7 additions for a 6th order polynomial. (The streamlined form was not used in the simulations reported here.) The calculation of $P_n(\chi^2)$ may also be done in a subroutine which is used for all trigonometric calculations in the airborne computer.

REFERENCES

1. Wilcox, J. C.: A New Algorithm for Strapped-Down Inertial Navigation. IEEE Transactions on Aerospace and Electronic Systems, vol. AES-3, no. 5, September 1967.

2. Crandall, S. H. et al: Dynamics of Mechanical and Electromechanical Systems. McGraw-Hill Book Company, New York, 1968.

3. United Aircraft Corporate Systems Center: A Study of the Critical Computational Problems Associated with Strapdown Inertial Navigational Systems. NASA CR-968, April 1968.

4. Sullivan, J. J.: Evaluation of the Computational Errors of Strapdown Navigational Systems. AIAA Journal, vol. 6, no. 2, February 1968.

5. Jordan, J. W.: Direction Cosine Computational Error. NASA TR-R-304, March 1969.

6. TRW Systems: Body-Fixed, Three-Axis Reference System Study, Phase II, Final Report. Prepared for NASA George C. Marshall Space Flight Center, Huntsville, Alabama, under Contract No. NAS8-20209, 15 December 1966.

7. Morgan, A. M.: Computational Errors in the Generation of the Direction Cosine Matrix in a Strapdown System. Journal of the Institute of Navigation. vol. 14, no. 1, Spring, 1967.

8. Hart, J. F. et al.: Computer Approximations. John Wiley and Sons, New York, 1968.